AMENDMENTS TO THE SPECIFICATION:

Please REPLACE the paragraph beginning at page 13, line 11, with the following rewritten paragraph:

-- Turning now to FIG. 4, a component block diagram of an example retail program 230 is illustrated. The program 230 typically comprises an application program module 232. The system 230 further comprises an executable program 234 inside of the application program module 232. An executable program is a program that can be run and typically means a compiled program translated into machine code in a format that can be loaded into memory and run by a computer's processor. The program 230 may also contain a module 236 that is executed by the executable program 234 inside the application program module 232. The program 230 also contains an exception filter 238. Exception filters are well-known in the art and may be registered by program modules when the program is started. When a failure or crash occurs, the exception filter code is executed. For example, suppose a failure occurs while executable program 234 is executing instructions running module 236. If executable program 234 has registered exception filter 238 with the operating system, then the exception filter ~~220~~ 238 is executed when executable program 234 encounters a failure. --

Please REPLACE the paragraph beginning at page 15, line 7, with the following rewritten paragraph:

-- Returning to FIG. 5, ~~I~~if the program 230 now crashes, as indicated by 260, the exception filter 238 will execute and close the program 230, as shown at 262. The log file 246 is also closed if there is a crash, as shown at 264. The failure reporting executable 240 is then executed

to report the crash to the repository 242. Failure reporting executable 240 is given instructions to locate and attach the log file 246, as shown at 266 and 268, respectively. Therefore, log file 246 will be sent to the repository 242, along with any other information gathered by failure reporting executable 240, as shown at 270. The crash data is sent to the repository 242 as .cab files, or in some other known compressed format. .cab is used as a file extension for cabinet files. Cabinet files are multiple files compressed into one, and later extractable with an extract.exe utility. It should be understood that while it has been described as using failure reporting module 240 to locate and attach the log file 246 and send it to the repository 242, any other of a number of ways to send the log file to the developer could be used and are within the scope of this invention. If the program closes normally, without crashing, the log file 246 is simply deleted. --

Please REPLACE the paragraph beginning at page 16, line 10, with the following rewritten paragraph:

-- Turning now to FIG. 7, the process of analyzing the data from the log file 246 is illustrated. The process begins at 274 with the developer obtaining the crash information from the repository 242. The developer the opens the .cab files containing the crash information at 276. The log file 246 is processed separately from any other crash data the .cab files may contain. Therefore, the log file 246 is shown separately in FIG. 7. A regeneration tool 278 combines the log file 246 with the string file 222 that was separated during the build process. Regeneration tool 278 locates the string file 222, by version number, that matches the version number of the program generating the log file 246, and maps the tag of each event to its corresponding string. In other words, regeneration tool 278 takes the compact event from log file 246 and makes it more meaningful by

regenerating the textual string previously removed. This process is essentially a decoding process, where the tags are converted back into meaningful text. Continuing with the example above, the regeneration tool displays the event as "Name lookup failed, 123." The combined result of the regeneration tool 278 is a text document output, shown as 280. By examining this text document, the developer can determine the events that were taking place prior to the crash, as shown at 282. The developer can use the output 280, along with other debugging tools and processes 284 to diagnose the crash and develop a solution 286. --